

String Vector based AHC Variants for Text Clustering

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we modify the AHC variants into the string vector-based versions and apply them to the text clustering. The initial AHC version was previously modified into the string vector-based versions, as an approach to the task. In this research, in the case of the numerical vector-based versions, we mention the three AHC variants: one where the data clustering proceeds in the bottom-up direction with the similarity threshold, one where it allows the merge of more than two pairs, and one where the clusters are merged based on the radius. The three AHC variants are modified into the string vector-based versions as the approaches to the text clustering, as well as the initial AHC algorithm. The goal of this research is to improve the clustering performance by proposing the modified versions of the AHC variants.

I. INTRODUCTION

The text clustering is defined as the process of generating several subgroups which are called clusters from a single text group, depending on text contents. In this task, an entire group of texts is given as the input, and each subgroup in the clustering results. The process of text clustering is to encode texts into structured data, define a similarity metric between text representations, and apply an unsupervised learning algorithm to this task. The task of this research is the hard clustering where each text is allowed to belong to only one cluster, and this task is expanded into the soft text clustering and the hierarchical text clustering in subsequent research. Before clustering texts, the outlier detection in the text group may be considered for cleaning the clustering results.

The motivation is provided for this research by the previous work which proposed the string vector based AHC algorithm as the approach to the text clustering. Encoding texts into string vectors and defining the similarity metric between string vectors as the basic semantic operation on them is the solution to the main problems in encoding texts into numerical vectors. The AHC algorithm was modified into the version which processes string vectors directly by the basic semantic operation on string vectors. The string vector-based version of AHC algorithm was empirically validated as its better performance than the traditional version in the text clustering. The upgrading points of the previous work are to modify the AHC variants into the string vector-based versions and to apply them to the text clustering.

The idea of this research is to modify the AHC variants into the string vector-based versions and apply them to the

text clustering. The semantic similarity between two string vectors was proposed for computing the similarity between two clusters. The three AHC variants are one where the data clustering proceeds in the bottom-up direction with the parameter, similarity threshold, one where it is allowed to merge more than one pair in each iteration, and one where clusters are merged in each iteration, depending on the radius from a particular cluster. The three AHC variants are modified into the string vector-based versions using the semantic similarity. The string vector-based versions of the AHC variants are used for clustering texts, based on their contents.

Let us mention some benefits from this research. The problems which are caused by encoding texts into numerical vectors are solved by encoding them into string vectors. It is intended to replace the AHC algorithm by the AHC variants for improving the clustering speed. It is intended to modify the AHC variants into the string vector-based versions for improving the clustering performance. We propose the more recommendable approaches for implementing the text clustering system by improving the clustering speed and the clustering performance.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with the previous works which are relevant to this research. It is intended to expand the style of modifying the AHC algorithm to its three variants. The directions of exploring previous works are derivation of AHC variants, modification of the standard AHC algorithm into its string vector-based version, and the cases of encoding a text into a string vector. The distinguishable points of this research are derivation of three AHC variants from the standard version, modification of them into string vector-based versions, and their applications to the text clustering. This section is intended to explore previous works for providing the background for this research.

Let us explore the previous cases of deriving AHC variants. In 2012, Murtagh and Contreras mentioned the possibility of deriving various versions of AHC algorithm and a particular variant which uses nearest neighbors [3]. In 2013, Sasirekha and Baby presented various similarity metrics between vectors and strategies of defining a similarity between clusters [4]. In 2015, Nazari et al. proposed the AHC algorithm which merges two clusters based on their common nearest neighbors [6]. In the AHC algorithms which are mentioned in above previous literatures, only one pair of clusters is merged in each iteration.

Let us explore the previous works which are concerned with the application of the modified version of the standard AHC algorithm to text clustering. In 2017, Jo initially proposed that the string vector-based version should be applied to the text clustering [7]. In 2019, Jo validated empirically its performances in the text clustering [9]. In 2020, he published the journal article which describes the string vector-based AHC algorithm and its application to the text clustering for its publication [11]. What is provided by the previous works becomes the basis for this research.

Let us explore the previous works on the neural networks, NTC (Neural Text Categorizer), where encoding a text into a string vector was initially proposed. In 2010, it was initially invented as an approach to the text classification by Jo [1]. In 2015, it was applied to the topic identification of Arabic texts by Abainia [5]. In 2018, it was evaluated as the most practical neural networks by Lakshmi and Rao [8]. In 2010, the NTSO (Neural Text Self Organizer) was also invented as an approach to the text clustering by Jo [2].

Let us mention the differences of this research from the previous works which are explored above. In this research, we derive the AHC variants as fast clustering algorithms by allowing merging multiple cluster pairs for each iteration. The three AHC variants are modified into string vector-based versions and adopted for implementing the text clustering system. In this research, the string vector based AHC variants are developed as clustering algorithms as well as classification algorithms. We will describe the modified AHC variants in Section III.

III. PROPOSED WORKS

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a text into a string vector and the proposed similarity metric. In Section III-B, we describe the standard version of AHC algorithm where the proposed similarity metric is adopted. In Section III-C, we derive the three variants from the standard version. In Section III-D, we present the system architecture of the text clustering system.

A. Encoding Proccerss

This section is concerned with the string vector as the text representation, and the similarity between string vectors.

A string vector is a finite ordered set of strings; numerical values are replaced by strings as the elements in a vector. It is required to define the similarity matrix for performing the operation on string vectors. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a text into a string and the process of computing the similarity between string vectors.

The process of encoding a text into a string vector is illustrated in Figure 1. A text is indexed into a list of words and their information. The features of a string vector are defined as symbolic forms manually in the feature definition. A string which represents a text is filled with the words which corresponds to the features in the feature value assignment. Refer to [10] for studying the encoding process in more detail.

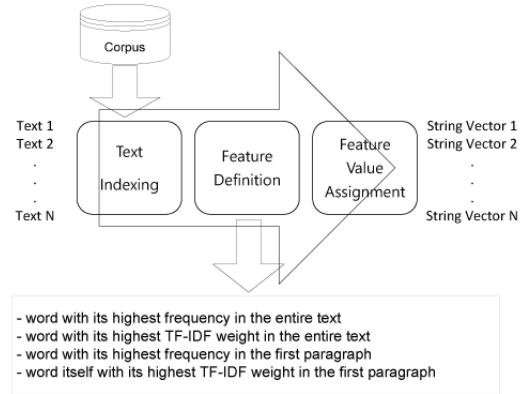


Figure 1. Process of Encoding Texts into String Vectors

The similarity matrix which is the basis for computing the similarity between two string vectors is illustrated in Figure 2. Each row and each column correspond to their own word, and each element in the similarity matrix is the similarity between two words, t_1 and t_2 , as shown in equation (1),

$$s_{ij} = \text{sim}(t_i, t_j) \quad (1)$$

The similarity between two words, t_i and t_j is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2\#(t_i \wedge t_j)}{\#(t_i) + \#(t_j)} \quad (2)$$

where $\#(t_i \wedge t_j)$ is the number of texts which include both t_i and t_j , $\#(t_i)$ is the number of texts which includes t_i , and $\#(t_j)$ is the number of texts which includes t_j . The value of similarity between two words, t_i and t_j , is always given as a normalized value between zero and one, as shown in equation (3),

$$0 \leq \text{sim}(t_i, t_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent texts as the reference table.

$$\begin{array}{c}
\begin{array}{cccc}
\text{Word 1} & \text{Word 2} & \dots & \text{Word N} \\
\text{Word 1} & \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1N} \end{bmatrix} \\
\text{Word 2} & \begin{bmatrix} S_{21} & S_{22} & \dots & S_{2N} \end{bmatrix} \\
\vdots & \vdots \\
\text{Word N} & \begin{bmatrix} S_{N1} & S_{N2} & \dots & S_{NN} \end{bmatrix}
\end{array}
\end{array}
\quad s_{ij} = \frac{2 \times \#(\text{word}_i, \text{word}_j)}{\#(\text{word}_i) + \#(\text{word}_j)}$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between texts. The two string vectors which represent texts are notated by $\mathbf{str}_1 = [t_{11} \ t_{12} \ \dots \ t_{1d}]$ and $\mathbf{str}_2 = [t_{21} \ t_{22} \ \dots \ t_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$sim(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d} \sum_{i=1}^d sim(t_{1i}, t_{2i}). \quad (4)$$

The similarity between elements, $sim(t_{i1}, t_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq sim(\mathbf{str}_1, \mathbf{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between two string vectors. The similarity between two string vectors is computed by equation (4). In the future research, we consider representing a text into multiple string vectors.

B. Initial Version of String Vector based AHC Algorithm

This section is concerned with the initial version of string vector based AHC algorithm. The version of AHC algorithm was initially proposed as the approach to the text clustering by Jo in 2019 [10]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between clusters. The AHC algorithm proceeds clustering data items with the bottom-up direction. This section is intended to describe the initial version of AHC algorithm from which its variants are derived.

The initial status for applying the AHC algorithm to the word clustering is illustrated in Figure 3. The words as clustering targets are encoded into string vectors by the process, which was described in Section III-A, and they are notated by a set, $Tr = \{\mathbf{str}_1, \mathbf{str}_2, \dots, \mathbf{str}_N\}$. The clusters are defined as many as data items, as $C_1, C_2, \dots, C_N$, and each cluster includes a data item as $C_i = \{\mathbf{str}_i\}$. The initial status which is represented in Figure 00 is notated by

$C_1 = \{\mathbf{str}_1\}, C_2 = \{\mathbf{str}_2\}, \dots, C_N = \{\mathbf{str}_N\}$. The cluster with only one item is called singleton, and singletons are constructed as many as data items in the initial status.



Figure 3. Initial Clusters in using AHC Algorithm: Singletons

The process of computing the similarity between two clusters is illustrated in Figure 4. The two clusters are notated by two sets of items, $C_1 = \{\mathbf{str}_{11}, \mathbf{str}_{12}, \dots, \mathbf{str}_{1|C_1|}\}$ and $C_2 = \{\mathbf{str}_{21}, \mathbf{str}_{22}, \dots, \mathbf{str}_{2|C_2|}\}$. The similarity between clusters, C_1 and C_2 , is computed by equation (6),

$$sim(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{i=1}^{|C_1|} \sum_{j=1}^{|C_2|} sim(\mathbf{str}_{1i}, \mathbf{str}_{2j}). \quad (6)$$

If the similarity metric which was described in Section III-A is adopted, the similarity between two clusters is always a normalized value between zero and one, as expressed in equation (7),

$$0 \leq sim(C_1, C_2) \leq 1. \quad (7)$$

The complexity of computing the similarity between two clusters is expressed by $O(|C_1||C_2|)$.

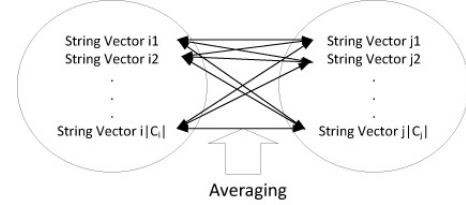


Figure 4. Similarity between Clusters

The process of clustering data items is illustrated as pseudo codes in Figure 5. The AHC algorithm begins with the status where singletons are given as many as data items. All possible pairs of clusters are generated from the current cluster list, their similarities are computed by equation (6), and the cluster pair with the highest similarity is merged into a cluster. In the AHC algorithm, the data items are clustered by iterating computing similarities of all possible cluster pairs and merge a cluster pair into a cluster. In each iteration, one cluster is decreased, and this iteration continues until reaching the desirable number of clusters.

Let us make some remarks on the initial version of the AHC algorithm from which we derive the variants. The initial status in applying the AHC algorithm to data clustering is that there are singletons as many as data items. The similarity between clusters is computed by averaging the similarities of all possible pairs from them. Data items are clustered by iterating computing the similarities of all

```

List clusterDataNumericalVectorList(List<String> vectorList, int desiredClusterListSize){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<Cluster> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }

    int clusterListSize = clusterList.size();

    //Cluster Data items in the bottom up direction
    while(clusterListSize > desiredClusterListSize){
        double maxSimilarity = 0.0;
        int maxIndex1 = 0;
        int maxIndex2 = 0;
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            for(int j = 0; j < clusterListSize; j++){
                Cluster c2 = clusterList.getElement(j);
                double similarity = sim(c1, c2);
                if(similarity > maxSimilarity){
                    maxSimilarity = similarity;
                    maxIndex1 = i;
                    maxIndex2 = j;
                }
            }
        }
        Cluster cmax1 = clusterList.getElement(maxIndex1);
        Cluster cmax2 = clusterList.getElement(maxIndex2);
        Cluster mergeCluster = cmax1.merge(cmax2);
        clusterList.deleteElement(cmax1);
        clusterList.deleteElement(cmax2);
        clusterList.addElement(mergeCluster);
    }
}

```

Figure 5. Initial Version of AHC Algorithm

possible cluster pairs and merge the pair with its maximal similarity into one cluster. In its variant, we will consider allow more than one pair of clusters to be merged.

C. AHC Variants

This section is concerned with the variants which are derived from the AHC algorithm which was covered in Section III-B. The difference from the traditional version of AHC algorithm is to adopt the similarity metric between string vectors for computing the similarity between clusters. In this section, we derive the three variants by merging cluster pairs by adopt the threshold-based selection, allowing merge of multiple pairs in each iteration, and merging clusters within the radius. In this research, we adopt the three variants of AHC algorithm for implementing the word clustering system. This section is intended to describe the three variants of AHC algorithm.

In Figure 6, we illustrate the process of clustering data items by the first variant of AHC algorithm. The clusters are initialized with singletons like the initial version which was described in Section III-B. All possible cluster pairs are generated, and the cluster pairs whose similarities are more than or equality to the similarity threshold are merged. If there is no pair which satisfies the condition, clustering data items is terminated. In this variant, the number of clusters is decreased variably in this variant.

We illustrate the process of clustering data items by the second variant of AHC algorithm in Figure 7 as a pseudo

```

List clusterDataNumericalVectorList(List<String> vectorList, double similarityThreshold){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<Cluster> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }

    boolean similarityFlag = false;
    //Cluster Data items in the bottom up direction
    while(similarityFlag){
        int clusterListSize = clusterList.size();
        similarityFlag = false;
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            for(int j = 0; j < clusterListSize; j++){
                Cluster c2 = clusterList.getElement(j);
                double similarity = sim(c1, c2);
                if(similarity >= similarityThreshold){
                    Cluster mergeCluster = c1.merge(cmax2);
                    clusterList.updateElement(i, mergeCluster);
                    clusterList.deleteElement(j);
                    clusterListSize = clusterList.size();
                    similarityFlag = true;
                }
            }
        }
    }
}

```

Figure 6. Feature Similarity based AHC Variant 1: Similarity Threshold Based AHC Algorithm

code. The number of cluster pairs which are merged into each iteration is given as an additional parameter in this variant. All possible cluster pairs are generated, and the similarity is computed for each pair. The cluster pairs are sorted by the descending order of the similarity, and the pairs within the rank are merged. The difference of this variant from the initial version is to merge multiple cluster pairs, instead of one pair.

```

List clusterDataNumericalVectorList(List<String> vectorList, double similarityThreshold){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<Cluster> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }

    int clusterListSize = clusterList.size();

    //Cluster Data items in the bottom up direction
    while(clusterListSize > desiredClusterSize){
        List<Pair> pairList = new List<>();
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            for(int j = i; j < clusterListSize; j++){
                Cluster c2 = clusterList.getElement(j);
                Pair indivPair = new Pair(c1, c2);
                indivPair.computeSimilarity();
                pairList.addElement(indivPair);
            }
        }
        pairList.sortPairs();
        List<Pair> selectedPairList = pairList.getSubList(0, rank-1);
        clusterList.mergeClusters(selectedPairList);
        clusterListSize = clusterList.size();
    }
}

```

Figure 7. Feature Similarity based AHC Variant 2: Merge of Multiple Pairs

In Figure 8, we illustrate the process of clustering data items by the last variant of AHC algorithm. The similarity threshold is given as an external parameter, and the number of other clusters whose similarity is greater than or equality to the similarity threshold is counted. In each iteration, the cluster with its maximal number of its similar clusters is appointed as the pivot, and the pivot cluster and its similar ones are merged into one cluster. This variant iterates selecting the pivot cluster in the current cluster list and merge the pivot and its similar ones, as the process of clustering data items. If there is no cluster with its similar cluster, the iteration is terminated.

```

List clusterDataNumericalVectorList(List<StringVectorList, double similarityThreshold>){
    int itemSize = stringVectorList.size();

    if(itemSize <= desiredClusterNumber)
        return null;
    List<ClusterList> clusterList = new List<>();
    //Initialize Clusters
    for(int i = 0; i < itemSize; i++){
        Cluster cc = new Cluster();
        StringVector dataItem = stringVectorList.getElement(i);
        cc.addDataItem(dataItem);
        clusterList.addElement(cc);
    }
    //Cluster Data items in the bottom up direction
    while(true){
        int clusterListSize = clusterList.size();
        if(clusterListSize == 1)
            return;
        int maxSimilarClusterSize = 0;
        int maxIndex = 0;
        for(int i = 0; i < clusterListSize; i++){
            Cluster c1 = clusterList.getElement(i);
            int similarClusterSize = c1.countSimilarClusters(clusterList);
            if(similarClusterSize > maxSimilarClusterSize){
                maxSimilarClusterSize = similarClusterSize;
                maxIndex = i;
            }
        }
        if(maxSimilarClusterSize == 0)
            return;
        Cluster pivotCluster = clusterList.getElement(maxIndex);
        List<ClusterList> similarClusterList = clusterList.getSimilarClusterList(pivotCluster);
        clusterList.mergeClusterList(similarClusterList);
    }
}

```

Figure 8. Feature Similarity based AHC Variant 3: Radius based AHC Algorithm

Let us make some remarks on the three variants of AHC algorithm which are adopted as the approaches to the word clustering. In each iteration of the first variant, cluster pairs with their similarities which are greater than or equal to the threshold are merged. In each iteration of the second variant, a fixed number of cluster pairs with their highest similarities are merged. In each iteration of the third variant, a pivot cluster and its similar clusters are merged. We may derive more variants from the AHC algorithm by setting different policies on merging clusters.

D. System Architecture

This section is concerned with the architecture and the execution flow of the text clustering system. In Section III-C, we describe the string vector based AHC variants and adopt them for implementing the text clustering system. In the architecture of the text clustering system, which was previous proposed, the initial AHC algorithm which is

mentioned in Section III-B is replaced by its variants. The process of executing the system is to encode the texts into string vectors and cluster them. This section is intended to describe the text clustering system which is implemented in this study with respect to its architecture.

A group texts and numerical vectors is illustrated in Figure 9. All of texts are initially given at a time under the assumption of the offline clustering. The texts in the group are encoded into string vectors by the process which is described in Section III-A. They are clustered by the AHC algorithm which is described in Section III-C. The online clustering where texts are given as a stream will be considered in next research.

Text 1	SV 1
.	.
.	.
Text N	SV N

Figure 9. Preparation of Training Examples

The architecture of the text clustering system is illustrated in Figure 10. In the encoding module, texts as clustering targets are encoded into string vectors. The clustering module includes the similarity computation module which computes the similarities among the string vectors by the process which is described in Section III-A and clusters them by one of the AHC variants. The text clusters are generated as the final output from the system. The difference from the system architecture which was proposed in [10] is to replace the initial AHC algorithm by its variants.

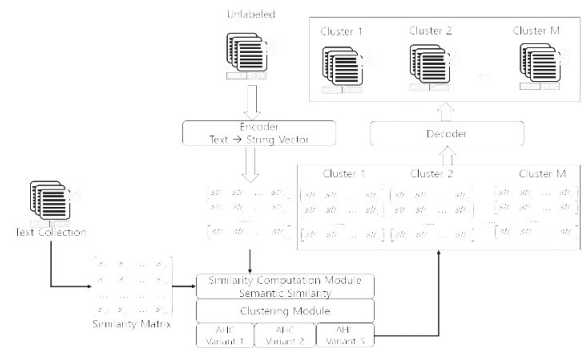


Figure 10. System Architecture

The execution flow of the text clustering system is illustrated in Figure 11. The texts are encoded into string vectors. The similarity metric between clusters which is expanded from one between string vectors is used in the AHC variants. The data items are clustered, selecting one among the three AHC variants which are described in Section III-C. The external parameter, the similarity threshold, is controlled in the first and the third variant for reaching the desired number of clusters.

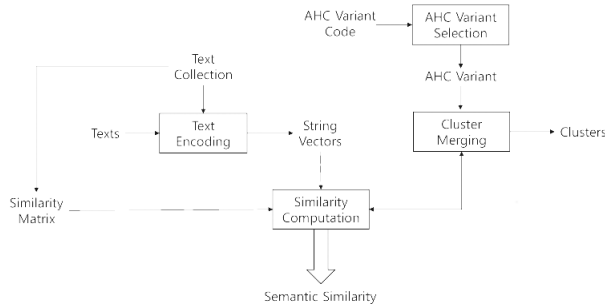


Figure 11. System Flow

Let us make some remarks on the proposed version of the text clustering system whose architecture is presented in Figure 10. The offline clustering is assumed in implementing the system; it will be expanded for doing online clustering in the next research. The upgrading point of the system is to replace the initial AHC algorithm by its variants which are described in Section III-C. The texts are encoded into string vectors, the similarity between clusters is computed by equation (6), and they are clustered by one of the AHC variants. It is expected that the clustering performance and speed are improved, by replacing the initial version by its variants.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We derive the three variants from the standard version by allowing multiple clusters to be merged at a time. We encode words into string vectors and apply the similarity metric among them to the AHC variants as well as the standard version. We design the text clustering system by adopt the AHC variants with the proposed similarity metric. In the next research, we will implement the text clustering system as a real system.

Let us mention the remaining tasks as the further discussions on this research. We may define other semantic operations on string vectors as well as the semantic similarity and characterize them mathematically. We modify other machine learning algorithms such as the Naïve Bayes and the SVM (Support Vector Machine) into the string vector-based versions. The proposed versions of the AHC variants may be applied to other clustering tasks. The text clustering may be implemented by adopting the proposed AHC variants as real programs.

REFERENCES

- [1] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", 83-96, International Journal of Information Studies, 2, 2, 2010.
- [2] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, Journal of Network Technology, 1, 1, 2010.

- [3] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview", 86-97, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2, 1, 2012.
- [4] K. Sasirekha and P. Baby. "Agglomerative hierarchical clustering algorithm-A Review", 83-85 International Journal of Scientific and Research Publications, 3,1, 2013.
- [5] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts." 1-8, The Proceedings of IEEE/ACS 12th International Conference of Computer Systems and Applications, 2015.
- [6] Z. Nazari, D. Kang, M.R. Asharif, Y. Sung, and S. Ogawa, "A new hierarchical clustering algorithm", 148-152, The Proceedings of IEEE International Conference on Intelligent Informatics and Biomedical Sciences, 2015.
- [7] T. Jo, "String Vector based AHC for Text Clustering", 673-677, The Proceedings of 18th International Conference on Advanced Communication Technology, 2017.
- [8] P. P. Lakshmi and D. R. Rao, "Text classification using artificial neural networks", 603-606, International Journal of Engineering & Technology, 7, 1, 2018.
- [9] T. Jo, "Introduction of String Vectors to AHC Algorithm for Clustering News Articles", 150-153, The Proceedings of 21st International Conference on Artificial Intelligence, 2019.
- [10] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.
- [11] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2020.